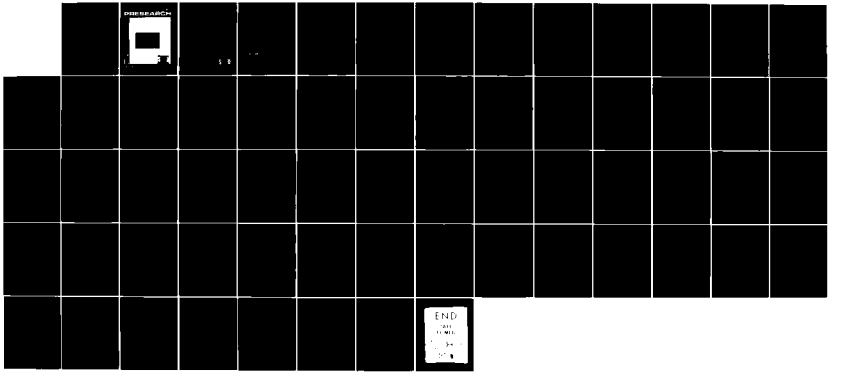
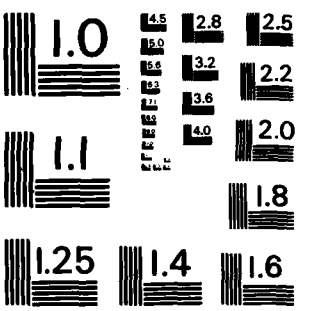
AD-A127 205 FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION VOLUME 6 1/1
SOURCE LISTINGS:..(U) PRESEARCH INC ARLINGTON VA
DEC 82 FPO-1-82-(37) N62477-81-C-0025

UNCLASSIFIED F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
10.0
11.2
12.5
14.0
16.0
18.0
20.0
22.4
25.0
28.0
32.0
36.0
40.0
45.0
50.0
56.0
63.0
71.0
80.0
90.0
100.0
112.0
125.0
140.0
160.0
180.0
200.0
224.0
250.0
280.0
320.0
360.0
400.0
450.0
500.0
560.0
630.0
710.0
800.0
900.0
1000.0
1120.0
1250.0
1400.0
1600.0
1800.0
2000.0
2240.0
2500.0
2800.0
3200.0
3600.0
4000.0
4500.0
5000.0
5600.0
6300.0
7100.0
8000.0
9000.0
10000.0
11200.0
12500.0
14000.0
16000.0
18000.0
20000.0
22400.0
25000.0
28000.0
32000.0
36000.0
40000.0
45000.0
50000.0
56000.0
63000.0
71000.0
80000.0
90000.0
100000.0
112000.0
125000.0
140000.0
160000.0
180000.0
200000.0
224000.0
250000.0
280000.0
320000.0
360000.0
400000.0
450000.0
500000.0
560000.0
630000.0
710000.0
800000.0
900000.0
1000000.0
1120000.0
1250000.0
1400000.0
1600000.0
1800000.0
2000000.0
2240000.0
2500000.0
2800000.0
3200000.0
3600000.0
4000000.0
4500000.0
5000000.0
5600000.0
6300000.0
7100000.0
8000000.0
9000000.0
10000000.0
11200000.0
12500000.0
14000000.0
16000000.0
18000000.0
20000000.0
22400000.0
25000000.0
28000000.0
32000000.0
36000000.0
40000000.0
45000000.0
50000000.0
56000000.0
63000000.0
71000000.0
80000000.0
90000000.0
100000000.0
112000000.0
125000000.0
140000000.0
160000000.0
180000000.0
200000000.0
224000000.0
250000000.0
280000000.0
320000000.0
360000000.0
400000000.0
450000000.0
500000000.0
560000000.0
630000000.0
710000000.0
800000000.0
900000000.0
1000000000.0
1120000000.0
1250000000.0
1400000000.0
1600000000.0
1800000000.0
2000000000.0
2240000000.0
2500000000.0
2800000000.0
3200000000.0
3600000000.0
4000000000.0
4500000000.0
5000000000.0
5600000000.0
6300000000.0
7100000000.0
8000000000.0
9000000000.0
10000000000.0
11200000000.0
12500000000.0
14000000000.0
16000000000.0
18000000000.0
20000000000.0
22400000000.0
25000000000.0
28000000000.0
32000000000.0
36000000000.0
40000000000.0
45000000000.0
50000000000.0
56000000000.0
63000000000.0
71000000000.0
80000000000.0
90000000000.0
100000000000.0
112000000000.0
125000000000.0
140000000000.0
160000000000.0
180000000000.0
200000000000.0
224000000000.0
250000000000.0
280000000000.0
320000000000.0
360000000000.0
400000000000.0
450000000000.0
500000000000.0
560000000000.0
630000000000.0
710000000000.0
800000000000.0
900000000000.0
1000000000000.0
1120000000000.0
1250000000000.0
1400000000000.0
1600000000000.0
1800000000000.0
2000000000000.0
2240000000000.0
2500000000000.0
2800000000000.0
3200000000000.0
3600000000000.0
4000000000000.0
4500000000000.0
5000000000000.0
5600000000000.0
6300000000000.0
7100000000000.0
8000000000000.0
9000000000000.0
10000000000000.0
11200000000000.0
12500000000000.0
14000000000000.0
16000000000000.0
18000000000000.0
20000000000000.0
22400000000000.0
25000000000000.0
28000000000000.0
32000000000000.0
36000000000000.0
40000000000000.0
45000000000000.0
50000000000000.0
56000000000000.0
63000000000000.0
71000000000000.0
80000000000000.0
90000000000000.0
100000000000000.0
112000000000000.0
125000000000000.0
140000000000000.0
160000000000000.0
180000000000000.0
200000000000000.0
224000000000000.0
250000000000000.0
280000000000000.0
320000000000000.0
360000000000000.0
400000000000000.0
450000000000000.0
500000000000000.0
560000000000000.0
630000000000000.0
710000000000000.0
800000000000000.0
900000000000000.0
1000000000000000.0
1120000000000000.0
1250000000000000.0
1400000000000000.0
1600000000000000.0
1800000000000000.0
2000000000000000.0
2240000000000000.0
2500000000000000.0
2800000000000000.0
3200000000000000.0
3600000000000000.0
4000000000000000.0
4500000000000000.0
5000000000000000.0
5600000000000000.0
6300000000000000.0
7100000000000000.0
8000000000000000.0
9000000000000000.0
10000000000000000.0
11200000000000000.0
12500000000000000.0
14000000000000000.0
16000000000000000.0
18000000000000000.0
20000000000000000.0
22400000000000000.0
25000000000000000.0
28000000000000000.0
32000000000000000.0
36000000000000000.0
40000000000000000.0
45000000000000000.0
50000000000000000.0
56000000000000000.0
63000000000000000.0
71000000000000000.0
80000000000000000.0
90000000000000000.0
100000000000000000.0
112000000000000000.0
125000000000000000.0
140000000000000000.0
160000000000000000.0
180000000000000000.0
200000000000000000.0
224000000000000000.0
250000000000000000.0
280000000000000000.0
320000000000000000.0
360000000000000000.0
400000000000000000.0
450000000000000000.0
500000000000000000.0
560000000000000000.0
630000000000000000.0
710000000000000000.0
800000000000000000.0
900000000000000000.0
1000000000000000000.0
1120000000000000000.0
1250000000000000000.0
1400000000000000000.0
1600000000000000000.0
1800000000000000000.0
2000000000000000000.0
2240000000000000000.0
2500000000000000000.0
2800000000000000000.0
3200000000000000000.0
3600000000000000000.0
4000000000000000000.0
4500000000000000000.0
5000000000000000000.0
5600000000000000000.0
6300000000000000000.0
7100000000000000000.0
8000000000000000000.0
9000000000000000000.0
10000000000000000000.0
11200000000000000000.0
12500000000000000000.0
14000000000000000000.0
16000000000000000000.0
18000000000000000000.0
20000000000000000000.0
22400000000000000000.0
25000000000000000000.0
28000000000000000000.0
32000000000000000000.0
36000000000000000000.0
40000000000000000000.0
45000000000000000000.0
50000000000000000000.0
56000000000000000000.0
63000000000000000000.0
71000000000000000000.0
80000000000000000000.0
90000000000000000000.0
100000000000000000000.0
112000000000000000000.0
125000000000000000000.0
140000000000000000000.0
160000000000000000000.0
180000000000000000000.0
200000000000000000000.0
224000000000000000000.0
250000000000000000000.0
280000000000000000000.0
320000000000000000000.0
360000000000000000000.0
400000000000000000000.0
450000000000000000000.0
500000000000000000000.0
560000000000000000000.0
630000000000000000000.0
710000000000000000000.0
800000000000000000000.0
900000000000000000000.0
1000000000000000000000.0
1120000000000000000000.0
1250000000000000000000.0
1400000000000000000000.0
1600000000000000000000.0
1800000000000000000000.0
2000000000000000000000.0
2240000000000000000000.0
2500000000000000000000.0
2800000000000000000000.0
3200000000000000000000.0
3600000000000000000000.0
4000000000000000000000.0
4500000000000000000000.0
5000000000000000000000.0
5600000000000000000000.0
6300000000000000000000.0
7100000000000000000000.0
8000000000000000000000.0
9000000000000000000000.0
10000000000000000000000.0
11200000000000000000000.0
12500000000000000000000.0
14000000000000000000000.0
16000000000000000000000.0
18000000000000000000000.0
20000000000000000000000.0
22400000000000000000000.0
25000000000000000000000.0
28000000000000000000000.0
32000000000000000000000.0
36000000000000000000000.0
40000000000000000000000.0
45000000000000000000000.0
50000000000000000000000.0
56000000000000000000000.0
63000000000000000000000.0
71000000000000000000000.0
80000000000000000000000.0
90000000000000000000000.0
100000000000000000000000.0
112000000000000000000000.0
125000000000000000000000.0
140000000000000000000000.0
160000000000000000000000.0
180000000000000000000000.0
200000000000000000000000.0
224000000000000000000000.0
250000000000000000000000.0
280000000000000000000000.0
320000000000000000000000.0
360000000000000000000000.0
400000000000000000000000.0
450000000000000000000000.0
500000000000000000000000.0
560000000000000000000000.0
630000000000000000000000.0
710000000000000000000000.0
800000000000000000000000.0
900000000000000000000000.0
1000000000000000000000000.0
1120000000000000000000000.0
1250000000000000000000000.0
1400000000000000000000000.0
1600000000000000000000000.0
1800000000000000000000000.0
2000000000000000000000000.0
2240000000000000000000000.0
2500000000000000000000000.0
2800000000000000000000000.0
3200000000000000000000000.0
3600000000000000000000000.0
4000000000000000000000000.0
4500000000000000000000000.0
5000000000000000000000000.0
5600000000000000000000000.0
6300000000000000000000000.0
7100000000000000000000000.0
8000000000000000000000000.0
9000000000000000000000000.0
10000000000000000000000000.0
11200000000000000000000000.0
12500000000000000000000000.0
14000000000000000000000000.0
16000000000000000000000000.0
18000000000000000000000000.0
20000000000000000000000000.0
22400000000000000000000000.0
25000000000000000000000000.0
28000000000000000000000000.0
32000000000000000000000000.0
36000000000000000000000000.0
40000000000000000000000000.0
45000000000000000000000000.0
50000000000000000000000000.0
56000000000000000000000000.0
63000000000000000000000000.0
71000000000000000000000000.0
80000000000000000000000000.0
90000000000000000000000000.0
100000000000000000000000000.0
112000000000000000000000000.0
125000000000000000000000000.0
140000000000000000000000000.0
160000000000000000000000000.0
180000000000000000000000000.0
200000000000000000000000000.0
224000000000000000000000000.0
250000000000000000000000000.0
280000000000000000000000000.0
320000000000000000000000000.0
360000000000000000000000000.0
400000000000000000000000000.0
450000000000000000000000000.0
500000000000000000000000000.0
560000000000000000000000000.0
630000000000000000000000000.0
710000000000000000000000000.0
800000000000000000000000000.0
900000000000000000000000000.0
1000000000000000000000000000.0
1120000000000000000000000000.0
1250000000000000000000000000.0
1400000000000000000000000000.0
1600000000000000000000000000.0
1800000000000000000000000000.0
2000000000000000000000000000.0
2240000000000000000000000000.0
2500000000000000000000000000.0
2800000000000000000000000000.0
3200000000000000000000000000.0
3600000000000000000000000000.0
4000000000000000000000000000.0
4500000000000000000000000000.0
5000000000000000000000000000.0
5600000000000000000000000000.0
6300000000000000000000000000.0
7100000000000000000000000000.0
8000000000000000000000000000.0
9000000000000000000000000000.0
10000000000000000000000000000.0
11200000000000000000000000000.0
12500000000000000000000000000.0
14000000000000000000000000000.0
16000000000000000000000000000.0
18000000000000000000000000000.0
20000000000000000000000000000.0
22400000000000000000000000000.0
25000000000000000000000000000.0
28000000000000000000000000000.0
32000000000000000000000000000.0
36000000000000000000000000000.0
40000000000000000000000000000.0
45000000000000000000000000000.0
50000000000000000000000000000.0
56000000000000000000000000000.0
63000000000000000000000000000.0
71000000000000000000000000000.0
80000000000000000000000000000.0
90000000000000000000000000000.0
100000000000000000000000000000.0
112000000000000000000000000000.0
125000000000000000000000000000.0
140000000000000000000000000000.0
160000000000000000000000000000.0
180000000000000000000000000000.0
200000000000000000000000000000.0
224000000000000000000000000000.0
250000000000000000000000000000.0
280000000000000000000000000000.0
320000000000000000000000000000.0
360000000000000000000000000000.0
400000000000000000000000000000.0
450000000000000000000000000000.0
500000000000000000000000000000.0
560000000000000000000000000000.0
630000000000000000000000000000.0
710000000000000000000000000000.0
800000000000000000000000000000.0
900000000000000000000000000000.0
1000000000000000000000000000000.0
1120000000000000000000000000000.0
1250000000000000000000000000000.0
1400000000000000000000000000000.0
1600000000000000000000000000000.0
1800000000000000000000000000000.0
2000000000000000000000000000000.0
2240000000000000000000000000000.0
2500000000000000000000000000000.0
2800000000000000000000000000000.0
3200000000000000000000000000000.0
3600000000000000000000000000000.0
4000000000000000000000000000000.0
4500000000000000000000000000000.0
5000000000000000000000000000000.0
5600000000000000000000000000000.0
6300000000000000000000000000000.0
7100000000000000000000000000000.0
8000000000000000000000000000000.0
9000000000000000000000000000000.0
10000000000000000000000000000000.0
11200000000000000000000000000000.0
12500000000000000000000000000000.0
14000000000000000000000000000000.0
16000000000000000000000000000000.0
18000000000000000000000000000000.0
20000000000000000000000000000000.0
22400000000000000000000000000000.0
25000000000000000000000000000000.0
28000000000000000000000000000000.0
32000000000000000000000000000000.0
36000000000000000000000000000000.0
40000000000000000000000000000000.0
45000000000000000000000000000000.0
50000000000000000000000000000000.0
56000000000000000000000000000000.0
63000000000000000000000000000000.0
71000000000000000000000000000000.0
80000000000000000000000000000000.0
90000000000000000000000000000000.0
100000000000000000000000000000000.0
112000000000000000000000000000000.0
125000000000000000000000000000000.0
140000000000000000000000000000000.0
160000000000000000000000000000000.0
180000000000000000000000000000000.0
200000000000000000000000000000000.0
224000000000000000000000000000000.0
250000000000000000000000000000000.0
280000000000000000000000000000000.0
320000000000000000000000000000000.0
360000000000000000000000000000000.0
400000000000000000000000000000000.0
450000000000000000000000000000000.0
500000000000000000000000000000000.0
560000000000000000000000000000000.0
630000000000000000000000000000000.0
710000000000000000000000000000000.0
800000000000000000000000000000000.0
900000000000000000000000000000000.0
1000000000000000000000000000000000.0
1120000000000000000000000000000000.0
1250000000000000000000000000000000.0
1400000000000000000000000000000000.0
1600000000000000000000000000000000.0
1800000000000000000000000000000000.0
2000000000000000000000000000000000.0
2240000000000000000000000000000000.0
2500000000000000000000000000000000.0
2800000000000000000000000000000000.0
3200000000000000000000000000000000.0
3600000000000000000000000000000000.0
4000000000000000000000000000000000.0
4500000000000000000000000000000000.0
5000000000000000000000000000000000.0
5600000000000000000000000000000000.0
6300000000000000000000000000000000.0
7100000000000000000000000000000000.0
8000000000000000000000000000000000.0
9000000000000000000000000000000000.0
10000000000000000000000000000000000.0
11200000000000000000000000000000000.0
12500000000000000000000000000000000.0
14000000000000000000000000000000000.0
16000000000000000000000000000000000.0
18000000000000000000000000000000000.0
20000000000000000000000000000000000.0
22400000000000000000000000000000000.0
25000000000000000000000000000000000.0
28000000000000000000000000000000000.0
32000000000000000000000000000000000.0
36000000000000000000000000000000000.0
40000000000000000000000000000000000.0
45000000000000000000000000000000000.0
50000000000000000000000000000000000.0
56000000000000000000000000000000000.0
63000000000000000000000000000000000.0
71000000000000000000000000000000000.0
80000000000000000000000000000000000.0
90000000000000000000000000000000000.0
100000000000000000000000000000000000.0
112000000000000000000000000000000000.0
125000000000000000000000000000000000.0
140000000000000000000000000000000000.0
160000000000000000000000000000000000.0
180000000000000000000000000000000000.0
200000000000000000000000000000000000.0
224000000000000000000000000000000000.0
250000000000000000000000000000000000.0
280000000000000000000000000000000000.0
320000000000000000000000000000000000.0
360000000000000000000000000000000000.0
400000000000000000000000000000000000.0
450000000000000000000000000000000000.0
500000000000000000000000000000000000.0
560000000000000000000000000000000000.0
630000000000000000000000000000000000.0
710000000000000000000000000000000000.0
800000000000000000000000000000000000.0
900000000000000000000000000000000000.0
1000000000000000000000000000000000000.0
1120000000000000000000000000000000000.0
1250000000000000000000000000000000000.0
1400000000000000000000000000000000000.0
1600000000000000000000000000000000000.0
1800000000000000000000000000000000000.0
2000000000000000000000000000000000000.0
2240000000000000000000000000000000000.0
2500000000000000000000000000000000000.0
2800000000000000000000000000000000000.0
3200000000000000000000000000000000000.0
3600000000000000000000000000000000000.0
4000000000000000000000000000000000000.0
4500000000000000000000000000000000000.0
5000000000000000000000000000000000000.0
5600000000000000000000000000000000000.0
6300000000000000000000000000000000000.0
7100000000000000000000000000000000000.0
8000000000000000000000000000000000000.0
9000000000000000000000000000000000000.0
10000000000000000000000000000000000000.0
11200000000000000000000000000000000000.0
12500000000000000000000000000000000000.0
14000000000000000000000000000000000000.0
16000000000000000000000000000000000000.0
18000000000000000000000000000000000000.0
20000000000000000000000000000000000000.0
22400000000000000000000000000000000000.0
25000000000000000000000000000000000000.0
28000000000000000000000000000000000000.0
32000000000000000000000000000000000000.0
36000000000000000000000000000000000000.0
40000000000000000000000000000000000000.0
45000000000000000000000000000000000000.0
50000000000000000000000000000000000000.0
56000000000000000000000000000000000000.0
63000000000000000000000000000000000000.0
71000000000000000000000000000000000000.0
80000000000000000000000000000000000000.0
90000000000000000000000000000000000000.0
100000000000000000000000000000000000000.0
112000000000000000000000000000000000000.0
125000000000000000000000000000000000000.0
140000000000000000000000000000000000000.0
160000000000000000000000000000000000000.0
180000000000000000000000000000000000000.0
200000000000000000000000000000000000000.0
224000000000000000000000000000000000000.0
250000000000000000000000000000000000000.0
280000000000000000000000000000000000000.0
320000000000000000000000000000000000000.0
360000000000000000000000000000000000000.0
400000000000000000000000000000000000000.0
450000000000000000000000000000000000000.0
500000000000000000000000000000000000000.0
560000000000000000000000000000000000000.0
630000000000000000000000000000000000000.0
710000000000000000000000000000000000000.0
800000000000000000000000000000000000000.0
900000000000000000000000000000000000000.0
1000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000.0
1250000000000000000000000000000000000000.0
1400000000000000000000000000000000000000.0
1600000000000000000000000000000000000000.0
1800000000000000000000000000000000000000.0
2000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000.0
2500000000000000000000000000000000000000.0
2800000000000000000000000000000000000000.0
3200000000000000000000000000000000000000.0
3600000000000000000000000000000000000000.0
4000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000.0
5000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000.0
6300000000000000000000000000000000000000.0
7100000000000000000000000000000000000000.0
8000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000.0
12500000000000000000000000000000000000000.0
14000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000.0
25000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000000000.0
2240000000000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000000000.0
22400000000000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000000000.0
224000000000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000000000.

# PRESEARCH

AD A127205

FLEET MOORING LEG
DESIGN PROGRAM DOCUMENTATION

Volume 6

SOURCE LISTINGS:
COMPOUND LEG REVERSE SOLUTIONS
AND POSTPROCESSOR

FPO-1-82-(37)

December 1982

General Distribution

DTIC FILE COPY

DTIC
ELECTE
APR 25 1983
S D
E

PRESEARCH INCORPORATED
2361 S. JEFFERSON DAVIS HIGHWAY, ARLINGTON, VA. 22202 (703) 553-2700

This document has been approved for public release and sale; its distribution is unlimited.

83 04 07 020

PRESEARCH INCORPORATED

# FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION

## Volume 6

## SOURCE LISTINGS: COMPOUND LEG REVERSE SOLUTIONS AND POSTPROCESSOR

FPO-1-82-(37)

December 1982

General Distribution

Performed for
Ocean Engineering and Construction Project Office
Chesapeake Division
Naval Facilities Engineering Command
Washington, D.C. 20374
Under
Contract N62477-81-C-0025

This document has been approved for public release and sale; its distribution is unlimited.

DTIC
ELECTE
APR 25 1983
S
E

Presearch Incorporated
2361 South Jefferson Davis Highway
Arlington, Virginia 22202

PRESEARCH INCORPORATED

# FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION

## Volume 6

## SOURCE LISTINGS: COMPOUND LEG REVERSE SOLUTIONS AND POSTPROCESSOR


| | Section | Volume |
|---|---|---|
| I. | EQUILIBRIUM EQUATIONS REPORT | 1 |
| II. | USER DOCUMENTATION | 2 |
| III. | SUBROUTINE DESCRIPTIONS | 3 |
| IV. | SOURCE LISTINGS | |
| | Query, Preprocessor, and Simple Leg | 4 |
| | Compound Leg Basic Solution | 5 |
| | Compound Leg Reverse Solutions and Postprocessor | 6 |
| | Table and Graphs | 7 |
| V. | LINK COMMAND FILES | 8 |
| VI. | PROGRAM FILE DESCRIPTIONS | 8 |
| VII. | COMMON BLOCKS | 8 |
| VIII. | DIRECTORY LISTINGS | 8 |
| IX. | IPL TAPE LISTINGS | 8 |



Accession For
NTIS GRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

DTIC COPY INSPECTED

# IV. SOURCE LISTINGS

## COMPOUND LEG REVERSE SOLUTIONS AND POSTPROCESSOR

CSEHP – CTEN3, PAGES 165 – 221

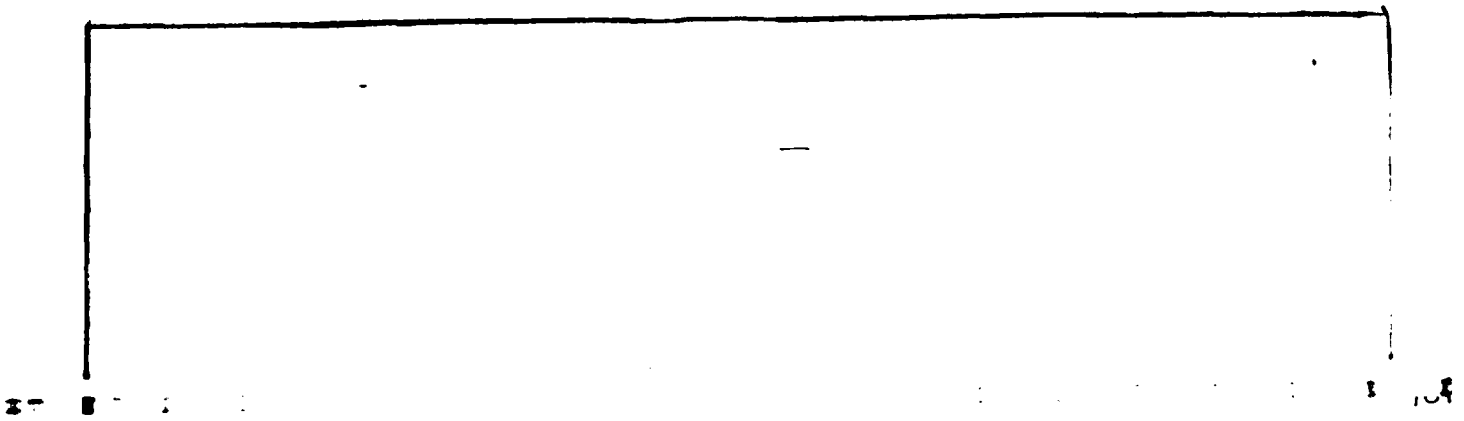

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MOOR01 | 1 | SEC1V | 65 | JUNCT | 127 | CTEN2 | 217 |
| MOOR02 | 3 | VCRITØ | 67 | EBUOY | 129 | CTEN3 | 218 |
| BKDAT | 5 | ESTV | 68 | SCOIL | 132 | GRAPH1 | 222 |
| QUERY | 6 | LENS | 69 | XSECV | 135 | GRIN1 | 224 |
| GEINIT | 17 | VFUN | 70 | SHIFT | 142 | GRIN2 | 228 |
| RW | 18 | CALC2 | 71 | SUBVX | 143 | GROUT1 | 233 |
| ADDEXT | 19 | EPSLV | 76 | WGTH | 145 | GROUT2 | 237 |
| ECHO | 20 | RDBACK | 80 | CERISR | 146 | GRAPH3 | 242 |
| OUTVAR | 25 | ELV1 | 83 | SECVIT | 148 | STICK | 244 |
| CONVRT | 27 | GCOEFF | 85 | STEFAB | 150 | SINKER | 247 |
| RWCOM1 | 32 | CTEN1 | 87 | CALC3 | 159 | ANCHOR | 249 |
| HXORV | 34 | CSLACK | 90 | EDGPT | 162 | ELIZER | 251 |
| SOLVE | 36 | CPREPØ | 94 | CSEHP | 165 | BUOY | 253 |
| PRSLV | 37 | CPREP1 | 97 | FTEN | 170 | ELVPNT | 255 |
| SECNT | 44 | SUMSC | 99 | CSSPR | 172 | SYMSNK | 265 |
| NWGT | 45 | CPREP2 | 100 | CSEPR | 177 | WELVPT | 266 |
| UMAP | 46 | CPREP3 | 102 | CSSXZ1 | 181 | ELVCAT | 267 |
| CUMAP | 47 | CSSHP | 104 | CSSXZ2 | 184 | PLNPNT | 270 |
| ISORT | 48 | PHIAB | 110 | CSEXZ1 | 188 | PLNSNK | 279 |
| TAUT | 49 | HSPLIT | 111 | CSEXZ2 | 191 | WPLNPT | 280 |
| STEF2A | 51 | TRISR | 113 | CEPSLV | 195 | PLNCAT | 281 |
| CHS2A | 54 | SCATAB | 115 | CRDBAK | 207 | MOOR04 | 283 |
| SEC1A | 55 | SRISR | 117 | COMMAP | 210 | HXCALC | 285 |
| CHS1A | 57 | TAN1 | 119 | CONTYP | 211 | HXCLC1 | 287 |
| CALC1 | 58 | XYCALC | 120 | ENOD | 212 | HXCLC2 | 291 |
| SLACK | 59 | TSPLIT | 122 | LENH | 213 | CSXHP | 295 |
| STEF2V | 61 | JTEN | 125 | ELV2 | 214 | MOOR05 | 296 |

```
e! sys final/t2for/csehp for##
      subroutine CSEHP(istart)
*****************************************************************************
      implicit integer*2 (*)
      implicit double precision (a-z)

      integer*2 istart

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
```

```
      integer*2 nc(2)
      equivalence (nca,nc)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision delyk,twod,halfd,dsq
      common /VANCH/ delyk,twod,halfd,dsq

      integer*2 iscopa,iscopb,itana,itanb,it,is
      double precision epsy,gamma,se
      common /VCMPD/ epsy,gamma,se,iscopa,iscopb,itana,itanb,it,is
      integer*2 iscop(2)
      equivalence (iscopa,iscop)

      integer*2 itold
      double precision ss0,dten0,ss1,dten1,ss2,dten2,slp0,sa0,smin(2)
      common /VEQUAL/ ss0,dten0,ss1,dten1,ss2,dten2,slp0,sa0,smin,
     &  itold
      equivalence (smin(1),samin),(smin(2),sbmin)

      double precision sa,sb,ca,cb,vc0a(6),vc0b(6),
     &  eex0,eez0,eey0,a0,b0,phia0,phib0
      integer*2 icase
      common /VSPID/ sa,sb,ca,cb,vc0a,vc0b,
     &  eex0,eez0,eey0,a0,b0,phia0,phib0,
     &  icase

      integer*2 istr1,iscop1,iscops,nit,isec
*****************************************************************************
      istr1=istart
      if (z(iscopa) eq sa0-slp0) istr1=0

      z(iscopa)=sa0-slp0
      z(iscopb)=se-z(iscopa)
      call CPREP1
      call CSSHP
      it=1
      if (ta le tb) it=2
      is=3-it
      call FTEN(dten0)
```

```
      if (dten0 le zero) goto 2000

      iscop1=iscop(it)
      iscops=iscop(is)
      ssmax=z(iscops)
      ssmin=smin(is)
      if (ssmax eq ssmin) goto 2000

      if (it ne itold  or  ss1 eq ssmin) istrt=0
      if (istrt eq 0) goto 100
      ss0=ss2
      z(iscops)=ss0
      z(iscop1)=se-ss0
      call CPREP1
      call CSSHP
      call FTEN(dten0)
      ss1=0.9999d0*ss0+0.0001d0*ssmax
      goto 150
  100 continue
      ss0=ssmax
      ss1=ssmin
      if (ssmin gt zero) ss1=half*(ssmin+ssmax)
  150 continue
*     write(10,*) it,is,iscop,se,ssmin
*     write(10,*) ss0,ta,tb,gamma,dten0,isol,ibrnch
      z(iscops)=ss1
      z(iscop1)=se-ss1
      call CPREP1
      call CSSHP
      call FTEN(dten1)
*     write(10,*) ss1,ta,tb,gamma,dten1,isol,ibrnch
      if (istrt ne 0) goto 500
      if (ss1 eq ssmin  and  dten1 gt zero) goto 2000

  300 continue
      if (dten1 le zero) goto 500
      ss1=half*(ss1+ssmin)
      z(iscops)=ss1
      z(iscop1)=se-ss1
      call CPREP1
      call CSSHP
      call FTEN(dten1)
*     write(10,*) ss1,ta,tb,gamma,dten1,isol,ibrnch
```

```
      goto 300

  500 continue
      ctot=ca+cb+c3+s4*w4
      epsdt=ctot*1.0d-8
      isec=3*istrt
      nit=1

 1000 continue
      ss2=half*(ss0+ss1)
      if (isec.eq.0  .and.  dten0-dten1.gt.ctot) goto 1020
      isec=isec+1
      ss2=ss1-dten1*(ss1-ss0)/(dten1-dten0)
      ssm=ssmin
      if (ssmin.gt.zero) ssm=half*(ssmin+dmin1(ss0,ss1))
      ss2=dmin1(ssmax,dmax1(ss2,ssm))
 1020 continue
      z(iscops)=ss2
      z(iscopt)=se-ss2
      call CPREP1
      call CSSHP
      call FTEN(dten2)
*     write(10,*) ss2,ta,tb,gamma,dten2,isol,ibrnch
      if (dabs(dten2).lt.epsdt) goto 2000
      if (nit.eq.30) stop 111
      if (isec.gt.3) goto 1250
      if (dten2.lt.zero) goto 1260
      ss0=ss2
      dten0=dten2
      goto 1300
 1250 continue
      ss0=ss1
      dten0=dten1
 1260 continue
      ss1=ss2
      dten1=dten2
 1300 continue
      nit=nit+1
      goto 1000

 2000 continue
      slp=se0-z(iscops)
      itold=it
```



```
      return
      end
```

```
e! sys final/t2for/ften for##
      subroutine FTEN(delten)
*************************************************************************
      implicit double precision (a-z)

      double precision delten

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
      double precision tjun(2)
```

```
      equivalence (ia,ijun)

      integer*2 iscopa,iscopb,iianb,iianb,ii,is
      double precision epsy,gamma,se
      common /VCMPD/ epsy,gamma,se,iscopa,iscopb,iianb,iianb,ii,is
*****************************************************************************
      delen=ijun(ii)-gamma*ijun(is)
      return
      end
*
```

```
e! sys final/12for/csspr for##
      subroutine CSSPR(ieps)
*************************************************************************
      implicit integer*2 (#)
      implicit double precision (a-z)

      integer*2 ieps

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
```

```
      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision delyk,twod,halfd,dsq
      common /VANCH/ delyk,twod,halfd,dsq

      double precision snphih,csphih,snafh,csafh,inafh,scafh,dsnph
      common /VHDIR/ snphih,csphih,snafh,csafh,inafh,scafh,dsnph

      double precision sa,sb,ca,cb,vc0a(6),vc0b(6),
     &  eex0,eez0,eey0,a0,b0,phia0,phib0
      integer*2 icase
      common /VSPID/ sa,sb,ca,cb,vc0a,vc0b,
     &  eex0,eez0,eey0,a0,b0,phia0,phib0,
     &  icase
      double precision vc0al
      equivalence (vc0a(1),vc0al)

      integer*2 ivs
      double precision v0,v1,v2,f0,f1,f2,f,eps
      common /VSEC/ v0,v1,v2,f0,f1,f2,f,eps,ivs

      integer*2 nit,i,ifail,ibis,isec,nwaa
      double precision zz(7)
      equivalence (zb(11),zz),(zb(18),scafz,halfdd,temp,sl,nwaa),
     &  (zb(19),csphin,hddcsp,rr),(zb(20),ssa)
*****************************************************************************
*     write(10,*) 'CSSPR',riot
      r=riot
      epsr=r*1.0d-10
      if (ieps.eq.1) epsr=r*1.0d-4

      scafz=SECNT(cz)
      csphin=csafh*(snphih+cz*inafh)/scafz
      halfdd=halfd*scafz
      hddcsp=halfdd*csphin
      temp=hddcsp*hddcsp-halfdd*halfdd
      sl=dmin1(dsqrt(sa*sa+temp)+hddcsp,dsqrt(sb*sb+temp)-hddcsp)

      do 100 i=1,7
```

```
            zz(i)=z(i)
100         continue

      ha=h
      s1a=s1
      w1a=(ca+cb)/s1
      c1a=c3
      s2a=s4
      w2a=w4
      call VCRIT0(2,za,vc0a)
      nwaa=0
      if (w1a lt zero  or  w2a lt zero) nwaa=1

      ssa=s1a+s2a
      h0=ca+cb+s4*w4
      f=do
      eps=h0*1 0d-10
      ivs=0
      ha=h0*1 0d-5
      call XSECV(2,za,vc0a,ssa,vc0a1,nwaa,ncb,zb,vc0b,
     &  snafh,csafh,tnafh,scafh,2,ifail)
      rr=dmax1(dmin1(r,0 01*xa+0 99*dsqrt(ssa*ssa-do*do)),xa)

      ha=h0
      call XSECV(2,za,vc0a,ssa,vc0a1,nwaa,ncb,zb,vc0b,
     &  snafh,csafh,tnafh,scafh,2,ifail)
      r0=xa
      h1=half*h0
      ha=h1
      call XSECV(2,za,vc0a,ssa,vc0a1,nwaa,ncb,zb,vc0b,
     &  snafh,csafh,tnafh,scafh,2,ifail)
      r1=xa

300   continue
      ha=h1-(r1-rr)*(h1-h0)/(r1-r0)
      if (ha le zero) ha=half*h1
      call XSECV(2,za,vc0a,ssa,vc0a1,nwaa,ncb,zb,vc0b,
     &  snafh,csafh,tnafh,scafh,2,ifail)
      h0=h1
      h1=ha
      r0=r1
      r1=xa
      if (dabs(r1-rr) gt eps) goto 300
```

```
      do 500 i=1,7
         z(i)=zz(i)
  500    continue
      call VCRIT0(nca,za,vc0a)

      h1=0.9d0*h0
*     write(10,*) h0,h1
      h=h0
      call CPREP3
      call CSSHP
      r0=rtot
*     write(10,*) h0,r0
      h=h1
      call CPREP3
      call CSSHP
      r1=rtot
*     write(10,*) h1,r1

      nit=1
      ibis=0
      isec=0
 1000 continue
      if (nit gt 15  and  (r0-r)*(r1-r) lt zero  and  ibis le 6
     &   and  (isec eq 0  or  isec gt 6)) goto 1020
      ibis=0
      isec=isec+1
      h=h1-(r1-r)*(h1-h0)/(r1-r0)
      if (h le zero) h=half*h1
      goto 1100
 1020 continue
      ibis=ibis+1
      isec=0
      h=half*(h0+h1)
 1100 continue
      call CPREP3
      call CSSHP
*     write(10,*) h,rtot
      if (dabs(rtot-r) lt epsr) goto 5000
      if (nit eq 100) stop 110
      if (ibis eq 0) goto 1350
      if ((rtot-r)*(r0-r) lt zero) goto 1370
```

```
      h0=h
      r0=rtot
      goto 1400
 1350 continue
      h0=h1
      r0=r1
 1370 continue
      h1=h
      r1=rtot
 1400 continue
      nit=nit+1
      goto 1000

 5000 continue
      return
      end
*
```

```
e: sys final/t2for/csepr for##
      subroutine CSEPR(ieps)
*************************************************************************
      implicit integer*2 (*)
      implicit double precision (a-z)

      integer*2 ieps

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
```

```
      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision delyk,twod,halfd,dsq
      common /VANCH/ delyk,twod,halfd,dsq

      integer*2 iscopa,iscopb,itana,itanb,it,is
      double precision epsy,gamma,se
      common /VCMPD/ epsy,gamma,se,iscopa,iscopb,itana,itanb,it,is
      integer*2 iscop(2)
      equivalence (iscopa,iscop)

      double precision snphih,csphih,snafh,csafh,tnafh,scafh,dsnph
      common /VHDIR/ snphih,csphih,snafh,csafh,tnafh,scafh,dsnph

      double precision sa,sb,ca,cb,vc0a(6),vc0b(6),
     &  eex0,eez0,eey0,a0,b0,phia0,phib0
      integer*2 icase
      common /VSPID/ sa,sb,ca,cb,vc0a,vc0b,
     &  eex0,eez0,eey0,a0,b0,phia0,phib0,
     &  icase
      double precision vc0al
      equivalence (vc0a(1),vc0al)

      integer*2 ivs
      double precision v0,v1,v2,f0,f1,f2,f,eps
      common /VSEC/ v0,v1,v2,f0,f1,f2,f,eps,ivs

      integer*2 nit,i,ibis,isec
*****************************************************************************
*     write(10,*) 'CSEPR',rtol
      r=rtol
      epsr=r*1 0d-10
      if (ieps eq 1) epsr=r*1 0d-4

      call CPREP1
      h1=(ca+cb+c3+s4*w4)*0 2d0
      h0=0 9d0*h1
      h=h0
      call CPREP3
```

```
*     write(10,*)
      call CSEHP(0)
      r0=rtot
*     write(10,*) h0,r0
      h=h1
      call CPREP3
*     write(10,*)
      call CSEHP(1)
      r1=rtot
*     write(10,*) h1,r1

      nit=1
      ibis=0
      isec=0
 1000 continue
      if (nit gt 15  and  (r0-r)*(r1-r) lt zero  and  ibis le 6
     &   and  (isec eq 0  or  isec gt 6)) goto 1020
      ibis=0
      isec=isec+1
      h=h1-(r1-r)*(h1-h0)/(r1-r0)
      if (h le zero) h=half*h1
      goto 1100
 1020 continue
      ibis=ibis+1
      isec=0
      h=half*(h0+h1)
 1100 continue
      call CPREP3
*     write(10,*)
      call CSEHP(1)
*     write(10,*) h,rtot
      if (dabs(rtot-r) lt epsr) goto 5000
      if (nit eq 100) stop 110
      if (ibis eq 0) goto 1350
      if ((rtot-r)*(r0-r) lt zero) goto 1370
      h0=h
      r0=rtot
      goto 1400
 1350 continue
      h0=h1
      r0=r1
 1370 continue
      h1=h
```

```
      r1=rtot
 1400 continue
      nit=nit+1
      goto 1000

 5000 continue
*     write(10,*) 'END CSEPR'
      return
      end
*
```

```
et sys final/t2for/cssxz1 for##
      subroutine CSSXZ1
*************************************************************************
      implicit integer*2 (*)
      implicit double precision (a-z)

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,a1a,va,s1a,w1a,c1a,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),a1a,va),
     &  (za(3),s1a),(za(4),w1a),(za(5),c1a),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,a1b,vb,s1b,w1b,c1b,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),a1b,vb),
     &  (zb(3),s1b),(zb(4),w1b),(zb(5),c1b),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
      double precision xztot(2)
      equivalence (xztot(1),xtot)
```

```
      double precision epsxz,xztru(2),xzbas(2),hbas(2),scratl(10)
      common /VCSSXZ/epsxz,xztru,xzbas,hbas,scratl
      double precision xtru,ztru,xbas,zbas,hbasx,hbasz
      equivalence (xztru(1),xtru),(xztru(2),ztru),
     &  (xzbas(1),xbas),(xzbas(2),zbas),
     &  (hbas(1),hbasx),(hbas(2),hbasz)
      double precision cospt,sinpt,hhx,hhz,dh0,dt0,dh1,dt1,dh2,dt2
      equivalence (scratl(1),cospt),(scratl(2),sinpt),
     &  (scratl(3),hhx),(scratl(4),hhz),
     &  (scratl(5),dh0),(scratl(6),dt0),
     &  (scratl(7),dh1),(scratl(8),dt1),
     &  (scratl(9),dh2),(scratl(10),dt2)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      integer*2 i
*****************************************************************************
*     write(10,*) 'CSSXZ1',xztot
      xtru=xtot
      ztru=ztot
      rtot=dsqrt(xtot*xtot+ztot*ztot)
      phih=datan(ztot/xtot)
      cospt= -ztot/rtot
      sinpt= xtot/rtot
      epsxz=rtot*rtot*1.0d-20

      call CPREP2
      call CSSPR(1)
      hhx=h*sinpt
      hhz= -h*cospt
*     write(10,*)
*     write(10,*) hhx,hhz,xztot

      hbasx=hhx
      hbasz=hhz
      xbas=xtot-xtru
      zbas=ztot-ztru
      dh0=zero
      dt0=xbas*cospt+zbas*sinpt
```

```
      if (d10*d10 le epsxz*1.0d12) goto 600

      do 500 i=1,5
            dh2= -h*d10/r1o1
            if (i gt 1) dh2=dh1-d11*(dh1-dh0)/(d11-d10)
            hbasx=hhx+dh2*cosp1
            hbasz=hhz+dh2*sinp1
            h=dsqrt(hbasx*hbasx+hbasz*hbasz)
            phih=datan(hbasz/hbasx)
            call CPREP2
            call CPREP3
            call CSSHP
*           write(10,*) hbas,xz1o1
            xbas=x1o1-x1ru
            zbas=z1o1-z1ru
            d12=xbas*cosp1+zbas*sinp1
            if (i eq 1) goto 300
            if (d12*d12 lt (xbas*xbas+zbas*zbas)*1.0d-2) goto 600
            dh0=dh1
            d10=d11
  300       continue
            dh1=dh2
            d11=d12
  500       continue

  600 continue
      return
      end
*
```

```
e! sys final/!2for/cssxz2 for##
      subroutine CSSXZ2
*************************************************************************
      implicit integer*2 (#)
      implicit double precision (a-z)

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
      double precision xztot(2)
      equivalence (xztot(1),xtot)
```

```
      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision epsxz,xziru(2),xzbas(2),hbas(2),scrat1(10)
      common /VCSSXZ/epsxz,xziru,xzbas,hbas,scrat1
      double precision xiru,ziru,xbas,zbas,hbasx,hbasz
      equivalence (xziru(1),xiru),(xziru(2),ziru),
     &  (xzbas(1),xbas),(xzbas(2),zbas),
     &  (hbas(1),hbasx),(hbas(2),hbasz)
      double precision dhmax,dhmin,hist(2),histx,histz
      equivalence (scrat1(1),dhmax,hist(1),histx),(dhmin,hist(2),histz)
      double precision delh(2),delhx,delhz,dhbx,dhbz
      equivalence (scrat1(3),delh(1),delhx,dhbx),(delh(2),delhz,dhbz)
      double precision jac(2,2),j11,j12,j21,j22
      equivalence (scrat1(5),jac(1,1),j11),(jac(1,2),j12),
     &  (jac(2,1),j21),(jac(2,2),j22)
      double precision jnew(4)
      equivalence (jac,jnew)
      double precision temp,xzsqo,detj
      equivalence (scrat1(9),temp,xzsqo),(scrat1(10),detj)

      double precision jold(4)
      integer*2 i,j,k,nit,iquit,ijo
*****************************************************************************
*     write(10,*)
*     write(10,*) 'CSSXZ2'

      iquit=0
      ijo=0
      nit=1
 1000 continue
*     write(10,*) hbas,xzbas
      xzsq=xbas*xbas+zbas*zbas
      if (iquit eq 1  or  xzsq le epsxz) goto 5000
      if (nit gt 10  and  xzsq le epsxz*1 0d4) goto 5000
      if (nit gt 5  and  xzsq le epsxz*1 0d8  and  xzsq ge xzsqo*half)
     &  goto 5000

      if (ijo eq 0  or  ijo gt 5) goto 1030
      ijo=ijo+1
```

```
      goto 1800
 1030 continue
      ijo=0
      if (init eq 1) goto 1050
      delhx=(j11*xbas+j12*zbas)*dsqrt(j21*j21+j22*j22)*detj
      delhz=(j21*xbas+j22*zbas)*dsqrt(j11*j11+j12*j12)*detj
*******************************************************************************
*     dhmax=h*1.0d-2
*     dhmin=h*1.0d-8
*     do 1020 i=1,2
*          delh(i)=dmax1(dhmin,dmin1(dhmax,delh(i)))
*1020      continue
*******************************************************************************
      goto 1100
 1050 continue
      delhx=h*1.0d-2
      delhz=delhx
 1100 continue

      do 1500 j=1,2
           hist(j)=hbas(j)+delh(j)
           k=3-j
           hist(k)=hbas(k)
           h=dsqrt(histx*histx+histz*histz)
           phih=datan(histz/histx)
           call CPREP2
           call CPREP3
           call CSSHP
           do 1400 i=1,2
                jac(i,j)=(xztot(i)-xztru(i)-xzbas(i))/delh(j)
 1400           continue
*          write(10,*) delh(j),xztot
 1500      continue

      if (init eq 1) goto 1625
      do 1620 i=1,4
           if (dabs(one-jnew(i)/jold(i)) gt 1.0d-2) goto 1625
 1620      continue
      ijo=1
 1625 continue
      do 1630 i=1,4
           jold(i)=jnew(i)
 1630      continue
```

```
      detj=j11*j22-j12*j21
*     write(10,*) jac,detj
*     write(10,*)
      if (nit gt 1  and  detj eq zero  and   xzsq le epsxz*1 0d8)
     &  iquit=1
      if (iquit eq 1) goto 2000
      temp=j11
      j11=j22/detj
      j22=temp/detj
      j12= -j12/detj
      j21= -j21/detj

 1800 continue
      dhbx=j11*xbas+j12*zbas
      dhbz=j21*xbas+j22*zbas
      temp=one
      if (dhbx+dhbx gt hbasx) temp=hbasx/(dhbx+dhb
      hbasx=hbasx-temp*dhbx
      hbasz=hbasz-temp*dhbz

 2000 continue
      h=dsqrt(hbasx*hbasx+hbasz*hbasz)
      phih=datan(hbasz/hbasx)
      call CPREP2
      call CPREP3
      call CSSHP
      xbas=xtot-xtru
      zbas=ztot-ztru
      xzsqo=xzsq
      nit=nit+1
      goto 1000

 5000 continue
      return
      end
*
```

```
et sys final/t2for/csexz1 for##
      subroutine CSEXZ1
*************************************************************************
      implicit integer*2 (#)
      implicit double precision (a-z)

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,a1a,va,s1a,w1a,c1a,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),a1a,va),
     &  (za(3),s1a),(za(4),w1a),(za(5),c1a),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,a1b,vb,s1b,w1b,c1b,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),a1b,vb),
     &  (zb(3),s1b),(zb(4),w1b),(zb(5),c1b),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
      double precision xztot(2)
      equivalence (xztot(1),xtot)
```

```
      double precision epsxz,xziru(2),xzbas(2),hbas(2),scrat1(10)
      common /VCSSXZ/epsxz,xziru,xzbas,hbas,scrat1
      double precision xiru,ziru,xbas,zbas,hbasx,hbasz
      equivalence (xziru(1),xiru),(xziru(2),ziru),
     &  (xzbas(1),xbas),(xzbas(2),zbas),
     &  (hbas(1),hbasx),(hbas(2),hbasz)
      double precision cospi,sinpi,hhx,hhz,dh0,di0,dh1,di1,dh2,di2
      equivalence (scrat1(1),cospi),(scrat1(2),sinpi),
     &  (scrat1(3),hhx),(scrat1(4),hhz),
     &  (scrat1(5),dh0),(scrat1(6),di0),
     &  (scrat1(7),dh1),(scrat1(8),di1),
     &  (scrat1(9),dh2),(scrat1(10),di2)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      integer*2 i
*****************************************************************************
*     write(10,*) 'CSEXZ1',xziot
      xiru=xiot
      ziru=ziot
      riot=dsqrt(xiot*xiot+ziot*ziot)
      phih=datan(ziot/xiot)
      cospi= -ziot/riot
      sinpi= xiot/riot
      epsxz=riot*riot*1.0d-20

      call CPREP2
      call CSEPR(1)
      hhx=h*sinpi
      hhz= -h*cospi
*     write(10,*)
*     write(10,*) hhx,hhz,xziot

      hbasx=hhx
      hbasz=hhz
      xbas=xiot-xiru
      zbas=ziot-ziru
      dh0=zero
      di0=xbas*cospi+zbas*sinpi
```

```
      if (d10*d10 le epsxz*1 0d12) goto 600

      do 500 i=1,5
           dh2= -h*d10/rtot
           if (i gt 1) dh2=dh1-d11*(dh1-dh0)/(d11-d10)
           hbasx=hhx+dh2*cospt
           hbasz=hhz+dh2*sinpt
           h=dsqrt(hbasx*hbasx+hbasz*hbasz)
           phih=datan(hbasz/hbasx)
           call CPREP2
           call CPREP3
*          write(10,*)
           call CSEHP(1)
*          write(10,*) hbas,xztot
           xbas=xtot-xtru
           zbas=ztot-ztru
           d12=xbas*cospt+zbas*sinpt
           if (i eq 1) goto 300
           if (d12*d12 lt (xbas*xbas+zbas*zbas)*1 0d-2) goto 600
           dh0=dh1
           d10=d11
  300      continue
           dh1=dh2
           d11=d12
  500      continue

  600 continue
      return
      end
*
```

```
e: sys final/t2for/csexz2 for##
      subroutine CSEXZ2
*************************************************************************
      implicit integer*2 (i-n)
      implicit double precision (a-z)

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,la,lb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,la,lb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)
      double precision xztot(2)
      equivalence (xztot(1),xtot)
```

```
      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision epsxz,xztru(2),xzbas(2),hbas(2),scrat1(10)
      common /VCSSXZ/epsxz,xztru,xzbas,hbas,scrat1
      double precision xtru,ztru,xbas,zbas,hbasx,hbasz
      equivalence (xztru(1),xtru),(xztru(2),ztru),
     &  (xzbas(1),xbas),(xzbas(2),zbas),
     &  (hbas(1),hbasx),(hbas(2),hbasz)
      double precision dhmax,dhmin,hist(2),histx,histz
      equivalence (scrat1(1),dhmax,hist(1),histx),(dhmin,hist(2),histz)
      double precision delh(2),delhx,delhz,dhbx,dhbz
      equivalence (scrat1(3),delh(1),delhx,dhbx),(delh(2),delhz,dhbz)
      double precision jac(2,2),j11,j12,j21,j22
      equivalence (scrat1(5),jac(1,1),j11),(jac(1,2),j12),
     &  (jac(2,1),j21),(jac(2,2),j22)
      double precision jnew(4)
      equivalence (jac,jnew)
      double precision temp,xzsqo,detj
      equivalence (scrat1(9),temp,xzsqo),(scrat1(10),detj)

      double precision jold(4)
      integer*2 i,j,k,nit,iquit,ijo
*****************************************************************************
*     write(10,*)
*     write(10,*) 'CSEXZ2'

      iquit=0
      ijo=0
      nit=1
 1000 continue
*     write(10,*)
*     write(10,*) 'ITER',nit
*     write(10,*) hbas,xzbas
      xzsq=xbas*xbas+zbas*zbas
      if (iquit eq 1  or  xzsq le epsxz) goto 5000
      if (nit gt 10  and  xzsq le epsxz*1 0d4) goto 5000
      if (nit gt 5  and  xzsq le epsxz*1 0d8  and  xzsq ge xzsqo*half)
     &  goto 5000
```

```
      if (ijo eq 0  or  ijo gt 5) goto 1030
      ijo=ijo+1
      goto 1800
 1030 continue
      ijo=0
      if (nit eq 1) goto 1050
      delhx=(j11*xbas+j12*zbas)*dsqrt(j21*j21+j22*j22)*detj
      delhz=(j21*xbas+j22*zbas)*dsqrt(j11*j11+j12*j12)*detj
*****************************************************************************
*     dhmax=h*1.0d-2
*     dhmin=h*1.0d-8
*     do 1020 i=1,2
*          delh(i)=dmax1(dhmin,dmin1(dhmax,delh(i)))
*1020      continue
*****************************************************************************
      goto 1100
 1050 continue
      delhx=h*1.0d-2
      delhz=delhx
 1100 continue

      do 1500 j=1,2
           hist(j)=hbas(j)+delh(j)
           k=3-j
           hist(k)=hbas(k)
           h=dsqrt(histx*histx+histz*histz)
           phih=datan(histz/histx)
           call CPREP2
           call CPREP3
*          write(10,*)
           call CSEHP(1)
           do 1400 i=1,2
                jac(i,j)=(xztot(i)-xztru(i)-xzbas(i))/delh(j)
 1400           continue
*          write(10,*) delh(j),xztot
 1500      continue

      if (nit eq 1) goto 1625
      do 1620 i=1,4
           if (dabs(one-jnew(i)/jold(i)) gt 1.0d-2) goto 1625
 1620      continue
      ijo=1
 1625 continue
```

```
      do 1630 i=1,4
           jold(i)=jnew(i)
 1630      continue

      detj=j11*j22-j12*j21
*     write(10,*)
*     write(10,*) jac,detj
      if (nit gt 1  and  detj eq zero  and   xzsq le epsxz*1 0d8)
     &  iquit=1
      if (iquit eq 1) goto 2000
      temp=j11
      j11=j22/detj
      j22=temp/detj
      j12= -j12/detj
      j21= -j21/detj

 1800 continue
      dhbx=j11*xbas+j12*zbas
      dhbz=j21*xbas+j22*zbas
      temp=one
      if (dhbx+dhbx gt hbasx) temp=hbasx/(dhbx+dhbx)
      hbasx=hbasx-temp*dhbx
      hbasz=hbasz-temp*dhbz

 2000 continue
      h=dsqrt(hbasx*hbasx+hbasz*hbasz)
      phih=datan(hbasz/hbasx)
      call CPREP2
      call CPREP3
*     write(10,*)
      call CSEHP(1)
      xbas=xtot-xtru
      zbas=ztot-ztru
      xzsqo=xzsq
      nit=nit+1
      goto 1000

 5000 continue
      return
      end
*
```

```
e! sys final/!2for/cepslv for##
      subroutine CEPSLV
*************************************************************************
      implicit integer*2 (*)
      implicit double precision (a-z)

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGL08/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
```

```
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision tnaf,phif
      common /VOFLR/ tnaf,phif

      double precision delyk,twod,halfd,dsq
      common /VANCH/ delyk,twod,halfd,dsq

      integer*1 ctitle(114)
      common /TITLES/ ctitle

      integer*1 cdatim(16)
      common /DATIME/ cdatim

      integer*1 cvarin(172)
      common /VARIN/ cvarin

      real lla,llb,ll,hha,hhb,hh,
     &  xx1a,xx3a,xx5a,xx1b,xx3b,xx5b,xx7,xx8,
     &  yy1a,yy3a,yy5a,yy1b,yy3b,yy5b,yy7,yy8,
     &  zz1a,zz3a,zz5a,zz1b,zz3b,zz5b,zz7,zz8,
     &  aa1a,aa2a,aa3a,aa4a,aa5a,aa6a,
     &  aa1b,aa2b,aa3b,aa4b,aa5b,aa6b,aa7,aa8,
     &  vv1a,vv2a,vv3a,vv4a,vv5a,vv6a,
     &  vv1b,vv2b,vv3b,vv4b,vv5b,vv6b,vv7,vv8,
     &  tt1a,tt2a,tt3a,tt4a,tt5a,tt6a,
     &  tt1b,tt2b,tt3b,tt4b,tt5b,tt6b,tt7,tt8,
     &  ddo,ddo,ddb,
     &  af,afdir,afa,adir,afb,bdir,
     &  sslp,coila,coilb
      integer*2 itsol,itbrn
      common /VAROUT/ lla,llb,ll,hha,hhb,hh,
     &  xx1a,xx3a,xx5a,xx1b,xx3b,xx5b,xx7,xx8,
     &  yy1a,yy3a,yy5a,yy1b,yy3b,yy5b,yy7,yy8,
     &  zz1a,zz3a,zz5a,zz1b,zz3b,zz5b,zz7,zz8,
     &  aa1a,aa2a,aa3a,aa4a,aa5a,aa6a,
     &  aa1b,aa2b,aa3b,aa4b,aa5b,aa6b,aa7,aa8,
     &  vv1a,vv2a,vv3a,vv4a,vv5a,vv6a,
     &  vv1b,vv2b,vv3b,vv4b,vv5b,vv6b,vv7,vv8,
     &  tt1a,tt2a,tt3a,tt4a,tt5a,tt6a,
     &  tt1b,tt2b,tt3b,tt4b,tt5b,tt6b,tt7,tt8,
```

```
     &  ddo,ddo,ddb,
     &  af,afdir,afo,adir,afb,bdir,
     &  ssip,coila,coilb,
     &  iisol,iibrn
      real parout(84)
      equivalence (iio,parout)

      integer*1 cvarg(240)
      common /VARG/ cvarg

      integer*1 cunkno(12)
      common /UNKNOW/ cunkno

      integer*1 cgropt(44)
      common /GROPT/ cgropt

      integer*1 cgrp21(218),cgrp22(82)
      common /GRP2CN/ cgrp21,cgrp22

      integer*2 i,nc,ioff,is,iy,ic,il,ip,iv
      dimension vc0(6,2),vc0a(6),vc0b(6),ang(6),ien(6),vien(6),yy(4)
      equivalence (vc0(1,1),vc0a),(vc0(1,2),vc0b)
*****************************************************************************
* BEGIN EXECUTABLE CODE
*****************************************************************************
* Read common blocks
*****************************************************************************
      call RWCOMI(1)

*****************************************************************************
      uz(3)=0
      call SUMSC(nca,za,sa,ca)
      call SUMSC(ncb,zb,sb,cb)
      call VCRIT0(nca,za,vc0a)
      call VCRIT0(ncb,zb,vc0b)

      csphih=dcos(phih)
      snphih=dsin(phih)
      inafh=dcos(phih-phif)*inaf
      scafh=SECNT(inafh)
      snafh=inafh/scafh
      csafh=one/scafh
```

```
*************************************************************************
* Initialize members of /VAROUT/ to 9999 99
*************************************************************************
      do 50 i=1,84
           parout(i)=9999 99
   50      continue

*************************************************************************
* Set branch index to zero if both branches are under tension
* Read branch index and solution type to /VAROUT/
*************************************************************************
      if (isol ne 1  and  isol ne 4) goto 60
      ibrnch=0
   60 continue
      iisol=isol
      iibrn=ibrnch


*************************************************************************
* Compute depths at anchors
* Read depths at origin and anchors to /VAROUT/
*************************************************************************
      da=do-half*delyk
      db=da+delyk
      ddo=do
      dda=da
      ddb=db

*************************************************************************
* Adjust slack lengths as necessary
*************************************************************************
      do 80 i=1,2
           if ( not  (isol eq 4  or  (isol eq 3  and  ibrnch eq i)))
     &       goto 80
           ioff=25*(i-1)
           il=ioff+24
           ip=il+1
           if (isol eq 4) goto 72
           z(ip)=phih
   72      continue
           hinf=h*dcos(z(ip)-phif)*inaf
           nc=(2-i)*nca+(i-1)*ncb
```

```
            lmax=zero
            is=ioff
            v=z(ioff+2)
            do 75 ic=1,nc
                 is=is+3
                 iv=is-ic
                 lmax=lmax+z(is)
                 if (v lt vc0(iv,i)+hinf  or  v ge vc0(iv-1,i)+hinf)
     &             goto 75
                 z(i1)=dmin1(z(i1),lmax)
   75            continue
   80       continue

*****************************************************************************
* Compute coil length, slack lengths, branch directions,
* and branch loads as necessary
* Compute effective ocean floor slopes for each branch
*****************************************************************************
       if (isol ne 2  and  isol ne 3) goto 140
       if (ibrnch ne 1) goto 105
       ut=one
       st=sa
       sc=sb
       goto 110
  105  continue
       ut= -one
       st=sb
       sc=sa
  110  continue
       zkt=ut*halfd
       ykt=cz*zkt
       if (isol ne 2) goto 115
       kte=st*csafh
       goto 120
  115  continue
       i=11+25*(ibrnch-1)
       kte=z(i)
  120  continue
       kcesq=(kte*csphih)**2+(zkt+zkt+kte*snphih)**2
       lc=dsqrt(kcesq+(ykt+ykt+kte*tnafh)**2)
       if (isol ne 2) goto 130
       coil=sc-lc
  130  continue
```

```
      phic=ut*(halfpi-dacos((dsq+kcesq-kte**2)/(twod*dsqrt(kcesq))))
      if (ibrnch ne 1) goto 135
      phia=phih
      if ( not  (isol eq 2  or  (isol eq 3  and  ib eq zero))) goto 132
      la=sa
  132 continue
      phib=phic
      lb=lc
      hb=zero
      goto 140
  135 continue
      phib=phih
      if ( not  (isol eq 2  or  (isol eq 3  and  ia eq zero))) goto 137
      lb=sb
  137 continue
      phia=phic
      la=lc
      ha=zero
  140 continue
      tnafa=dcos(phia-phif)*tnaf
      tnafb=dcos(phib-phif)*tnaf
      if ( not  (isol eq 1  or  (isol eq 2  and  ibrnch eq 1))) goto 145
      ha=ta/SECNT(tnafa)
  145 continue
      if ( not  (isol eq 1  or  (isol eq 2  and  ibrnch eq 2))) goto 150
      hb=tb/SECNT(tnafb)
  150 continue
      htnafa=ha*tnafa
      htnafb=hb*tnafb
      if (isol ne 1) goto 160
      la=sa
      lb=sb
  160 continue
      if (isol ne 3  and  isol ne 4) goto 170
      l=zero
  170 continue

*************************************************************************
* Read slack lengths, loads, ocean floor angles, equalizer slippage
* and coil length to /VAROUT/
*************************************************************************
      lla=la
      llb=lb
```

```
      ll=1
      hha=ha*1.0d-3
      hhb=hb*1.0d-3
      hh=h*1.0d-3
      af=datan(tnaf)*raddeg
      afdir=phif*raddeg
      afa=datan(tnafa)*raddeg
      adir=phia*raddeg
      afb=datan(tnafb)*raddeg
      bdir=phib*raddeg
      sslp=slp
      coila=0.0
      coilb=0.0
      if (isol.ne.2 .and. isol.ne.3) goto 190
      if (ibrnch.ne.1) goto 185
      coilb=coil
      goto 190
  185 continue
      coila=coil
  190 continue

*************************************************************************
* Compute branch component displacements and slopes as necessary
*************************************************************************
      if (isol.ne.1 .and. isol.ne.2) goto 220
      call FNOD(nca,za,la,tnafa)
      call FNOD(ncb,zb,lb,tnafb)
      goto 290
  220 continue
      if (isol.ne.3) goto 250
      if (ibrnch.ne.1) goto 225
      if (tb.ne.zero) goto 222
      call FNOD(nca,za,la,tnafa)
      goto 223
  222 continue
      za(2)=tnafa+dmax1(zero,(za(2)-vc0a(1)-htnafa)/ha)
  223 continue
      call FNOD(ncb,zb,lb,tnafb)
      lh=LENH(tb,ncb,zb)
      nc=ncb
      goto 230
  225 continue
```

```
      if (ia ne zero) goto 227
      call FNOD(ncb,zb,lb,tnafb)
      goto 228
  227 continue
      zb(2)=tnafb+dmax1(zero,(zb(2)-vc0b(1)-htnafb)/hb)
  228 continue
      call FNOD(nca,za,la,tnafa)
      lh=LENH(ia,nca,za)
      nc=nca
  230 continue
      ioff=25*(2-ibrnch)
      iy=ioff+16
      yy(1)= -yk1
      do 235 ic=1,nc
           yy(ic+1)=yy(ic)+z(iy)
           iy=iy+1
  235      continue
      ssum=zero
      is=ioff+3*nc
      iy=nc+1
      do 240 ic=1,nc
           yy(iy)=yy(iy)+dmax1(zero,lh-ssum)
           ssum=ssum+z(is)
           iy=iy-1
           is=is-3
  240      continue
      iy=ioff+16
      do 245 ic=1,nc
           z(iy)=yy(ic+1)-yy(ic)
           iy=iy+1
  245      continue
      goto 290
  250 continue
      za(2)=tnafa+dmax1(zero,(za(2)-vc0a(1)-htnafa)/ha)
      zb(2)=tnafb+dmax1(zero,(zb(2)-vc0b(1)-htnafb)/hb)
  290 continue

*************************************************************************
* Compute node coordinates and read into /VAROUT/
*************************************************************************
      snphia=dsin(phia)
      snphib=dsin(phib)
      csphia=dcos(phia)
```

```
      csphib=dcos(phib)
      hdelyk=half*delyk

      xx1a=0.0
      if (nca eq 1) goto 310
      xx3a=x1a*csphia
      if (nca eq 2) goto 310
      xx5a=(x1a+x2a)*csphia
  310 continue
      xx1b=0.0
      temp=x1b*csphib
      if (ncb eq 1) goto 315
      xx3b=temp
      temp=temp+x2b*csphib
      if (ncb eq 2) goto 315
      xx5b=temp
      temp=temp+x3b*csphib
  315 continue
      xx7=temp
      xx8=temp+x4*csphih

      zz1a=halfd
      if (nca eq 1) goto 320
      zz3a=halfd+x1a*snphia
      if (nca eq 2) goto 320
      zz5a=halfd+(x1a+x2a)*snphia
  320 continue
      zz1b= -halfd
      temp= -halfd+x1b*snphib
      if (ncb eq 1) goto 325
      zz3b=temp
      temp=temp+x2b*snphib
      if (ncb eq 2) goto 325
      zz5b=temp
      temp=temp+x3b*snphib
  325 continue
      zz7=temp
      zz8=temp+x4*snphih

      yy1a=hdelyk
      if (nca eq 1) goto 330
      yy3a=hdelyk+y1a
      if (nca eq 2) goto 330
```

```
      yy5a=hdelyk+(y1a+y2a)
  330 continue
      yy1b= -hdelyk
      temp= -hdelyk+y1b
      if (ncb eq 1) goto 335
      yy3b=temp
      temp=temp+y2b
      if (ncb eq 2) goto 335
      yy5b=temp
      temp=temp+y3b
  335 continue
      yy7=temp
      yy8=temp+y4

*************************************************************************
* Compute node angles and tensions and read into /VAROUT/
*************************************************************************
      if (isol eq 3  and  ibrnch eq 2) goto 410
      call CTEN3(nca,za,tnafa,ta,ang,ten,vten)
      goto 415
  410 continue
      call CTEN2(nca,za,vc0a,ta,tnafa,ang,ten,vten)
  415 continue
      aa1a=ang(1)
      t1a=ten(1)
      vv1a=vten(1)
      aa2a=ang(2)
      t2a=ten(2)
      vv2a=vten(2)
      if (nca eq 1) goto 420
      aa3a=ang(3)
      t3a=ten(3)
      vv3a=vten(3)
      aa4a=ang(4)
      t4a=ten(4)
      vv4a=vten(4)
      if (nca eq 2) goto 420
      aa5a=ang(5)
      t5a=ten(5)
      vv5a=vten(5)
      aa6a=ang(6)
      t6a=ten(6)
      vv6a=vten(6)
```

```
420 continue

    if (isol eq 3  and  ibrnch eq 1) goto 430
    call CTEN3(ncb,zb,tnafb,tb,ang,ten,vten)
    goto 435
430 continue
    call CTEN2(ncb,zb,vc0b,tb,tnafb,ang,ten,vten)
435 continue
    aa1b=ang(1)
    tt1b=ten(1)
    vv1b=vten(1)
    aa2b=ang(2)
    tt2b=ten(2)
    vv2b=vten(2)
    if (ncb eq 1) goto 440
    aa3b=ang(3)
    tt3b=ten(3)
    vv3b=vten(3)
    aa4b=ang(4)
    tt4b=ten(4)
    vv4b=vten(4)
    if (ncb eq 2) goto 440
    aa5b=ang(5)
    tt5b=ten(5)
    vv5b=vten(5)
    aa6b=ang(6)
    tt6b=ten(6)
    vv6b=vten(6)
440 continue

    ang7=datan(tana7)
    ang8=datan(tana8)
    if (l eq zero) goto 450
    ten7=(h*scafh-w4*l*snafh)*1 0d-3
    goto 455
450 continue
    ten7=h*SECNT(tana7)*1 0d-3
455 continue
    ten8=h*SECNT(tana8)*1 0d-3
    aa7=ang7*raddeg
    aa8=ang8*raddeg
    tt7=ten7
    tt8=ten8
```

```
      vv7=ten7*dsin(ang7)
      vv8=ten8*dsin(ang8)

*****************************************************************************
* Compute elevation view parameters
*****************************************************************************
      call ELV2(phih,tnafa,tnafb,tnafh,scafh,snafh,csafh)

*****************************************************************************
* Read solutions for unknown input parameters to /VARIN/
*****************************************************************************
      call CRDBAK(tnafa,tnafb)

*****************************************************************************
* Write to common blocks /VAROUT/ and /VARG/
*****************************************************************************
      call RWCOM1(2)

      return
      end
*
```

```
e! sys final/12for/crdbak for##
      subroutine CRDBAK(inafa,inafb)
*************************************************************************
      implicit double precision (a-z)

      double precision inafa,inafb

      integer*2 ileg,ist
      integer*4 nnca,nncb
      real angla,anglb,
     &  scopla,scoplb,wgtla,wgtlb,clmpla,clmplb,
     &  scop2a,scop2b,wgt2a,wgt2b,clmp2a,clmp2b,
     &  scop3a,scop3b,wgt3a,wgt3b,slip,fric1,clmp3,scop4,wgt4,anksep,
     &  plx,plz,pld,p2x,p2z,p2d,p3x,p3z,p3d,
     &  hload,hdir,rbuoy,xbuoy,zbuoy,deptho,pdir
      common /VARIN/ ileg,ist,nnca,nncb,angla,anglb,
     &  scopla,scoplb,wgtla,wgtlb,clmpla,clmplb,
     &  scop2a,scop2b,wgt2a,wgt2b,clmp2a,clmp2b,
     &  scop3a,scop3b,wgt3a,wgt3b,slip,fric1,clmp3,scop4,wgt4,anksep,
     &  plx,plz,pld,p2x,p2z,p2d,p3x,p3z,p3d,
     &  hload,hdir,rbuoy,xbuoy,zbuoy,deptho,pdir
      real parin(40)
      equivalence (angla,parin(1))

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,cla,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),cla),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,clb,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
```

```
     &   tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &   (zb(3),slb),(zb(4),wlb),(zb(5),clb),
     &   (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &   (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
     &   (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &   (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &   (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &   (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coil,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coil),(z(52),slp),(z(53),frct),(z(54),c3),
     &   (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &   (z(59),tana7),(z(60),tana8),(z(61),l),
     &   (z(62),h),(z(63),phih),
     &   (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &   izero,ione,itwo

      integer*2 i,i0,iz,itab,icon,commap,contyp,numax,u(5)
*************************************************************************
      numax=5
      i0=1
      do 20 i=1,numax
      u(i)=uz(i)
      if (u(i) ne 0) i0=i0+1
   20 continue
      if (ist ne 2) goto 50
      u(i0)=2
      u(i0+1)=27
   50 continue
      do 200 i=1,numax
      iz=u(i)
      if (iz eq 0) goto 200
      itab=COMMAP(iz)
      icon=CONTYP(iz)
      goto (110,120,130,140,150),icon
  110 continue
      parin(itab)=z(iz)
      goto 200
```

```
  120 continue
      parin(iiab)=z(iz)*1.0d-3
      goto 200
  130 continue
      parin(iiab)=(datan(z(iz))-datan(inafa))*raddeg
      goto 200
  140 continue
      parin(iiab)=(datan(z(iz))-datan(inafb))*raddeg
      goto 200
  150 continue
      parin(iiab)=z(iz)*raddeg
  200 continue
      xbuoy=xtot
      zbuoy=ztot
      depth0=d0

      return
      end
*
```

```
et sys final/t2for/commap for##
      function COMMAP(iz)
*************************************************************************
      integer*2 commap,iz
*************************************************************************
      if (iz lt 2  or  iz gt 10) goto 20
      commap=2*iz-3
      goto 100
   20 continue
      if (iz lt 27  or  iz gt 35) goto 30
      commap=2*iz-52
      goto 100
   30 continue
      if (iz lt 52  or  iz gt 56) goto 40
      commap=iz-33
      goto 100
   40 continue
      if (iz lt 62  or  iz gt 67) goto 100
      commap=iz-28
  100 continue

      return
      end
*
```

```
ed sys final/t2for/contyp for##
      function CONTYP(iz)
*****************************************************************************
      integer*2 contyp,iz
*****************************************************************************
      iiz=iz
      i1=(iiz-5)*(iiz-8)*(iiz-30)
      i2=(iiz-33)*(iiz-54)*(iiz-62)
      if (i1 ne 0  and  i2 ne 0) goto 30
      contyp=2
      goto 100
   30 continue
      if (iz ne 2) goto 40
      contyp=3
      goto 100
   40 continue
      if (iz ne 27) goto 50
      contyp=4
      goto 100
   50 continue
      if (iz ne 63) goto 60
      contyp=5
      goto 100
   60 continue
      contyp=1
  100 continue

      return
      end
*
```

```
e! sys final/i2for/fnod for##
      subroutine FNOD(nc,z,l,tanaf)
*****************************************************************************
* Compute node displacements and angles for single branch
* with junction on ocean floor
*****************************************************************************
      implicit double precision (a-z)

      integer*2 nc
      double precision z(25),l,tanaf

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      integer*2 ic,ia,is,ix,iy
*****************************************************************************
      cosaf=one/SECNT(tanaf)
      ssum=zero
      do 100 ic=1,nc
      ia=16+2*ic
      if (ic eq 1) ia=2
      z(ia)=tanaf
      is=3*ic
      s=dmax1(zero,dmin1(z(is),l-ssum))
      ix=12+ic
      z(ix)=s*cosaf
      iy=15+ic
      z(iy)=z(ix)*tanaf
      ia=17+2*ic
      z(ia)=tanaf
      ssum=ssum+z(is)
  100 continue
      return
      end
*
```

```
et sys final/t2for/lenh for##
      function LENH(ce,nc,z)
*************************************************************************
      implicit double precision (a-z)

      integer*2 nc
      double precision lenh,ce,z(25)

      integer*2 is
*************************************************************************
      is=3*nc
      wgt=z(is)*z(is+1)
      if (nc gt 1  and  ce gt wgt) goto 10
      lenh=ce/z(is+1)
      goto 100
   10 continue
      lenh=z(is)
      wgt=wgt+z(is-1)
      if (ce le wgt) goto 100
      if (nc gt 2  and  ce gt wgt+z(6)*z(7)) goto 30
      lenh=lenh+(ce-wgt)/z(is-2)
      goto 100
   30 continue
      lenh=lenh+z(6)
      wgt=wgt+z(6)*z(7)+z(5)
      if (ce le wgt) goto 100
      lenh=lenh+(ce-wgt)/z(4)
  100 continue
      return
      end
*
```

```
e! sys final/f2for/elv2 for##
      subroutine ELV2(phip,tnafa,tnafb,tnafh,scafh,snafh,csafh)
*************************************************************************
      implicit integer*2 (#)
      implicit double precision (a-z)

      double precision phip,tnafa,tnafb,tnafh,scafh,snafh,csafh

      double precision lla,llb,ll,tana,tanb,tanr,
     &  xx1a,xx3a,xx5a,xx3b,xx5b,xx7,xx8,
     &  ga11,ga12,ga21,ga22,ga31,ga32,
     &  gb11,gb12,gb21,gb22,gb31,gb32,
     &  g1,g2,xfa,xfb,xf
      common /VARG/ lla,llb,ll,tana,tanb,tanr,
     &  xx1a,xx3a,xx5a,xx3b,xx5b,xx7,xx8,
     &  ga11,ga12,ga21,ga22,ga31,ga32,
     &  gb11,gb12,gb21,gb22,gb31,gb32,
     &  g1,g2,xfa,xfb,xf

      integer*2 ileg,ist,nca,ncb,nwa,nwb,isol,ibrnch,uz(5)
      double precision z(67),cz,cx,d,ta,tb
      common /VGLOB/ ileg,ist,nca,ncb,z,cz,cx,d,ta,tb,nwa,nwb,
     &  isol,ibrnch,uz
      double precision za(25),zb(25)
      equivalence (z(1),za(1)),(z(26),zb(1))
      double precision ha,ala,va,sla,wla,c1a,s2a,w2a,c2a,s3a,w3a,
     &  xa,ya,x1a,x2a,x3a,y1a,y2a,y3a,
     &  tana2a,tana3a,tana4a,tana5a,tana6a,la,phia
      equivalence (za(1),ha),(za(2),ala,va),
     &  (za(3),sla),(za(4),wla),(za(5),c1a),
     &  (za(6),s2a),(za(7),w2a),(za(8),c2a),
     &  (za(9),s3a),(za(10),w3a),(za(11),xa),(za(12),ya),
     &  (za(13),x1a),(za(14),x2a),(za(15),x3a),
     &  (za(16),y1a),(za(17),y2a),(za(18),y3a),
     &  (za(19),tana2a),(za(20),tana3a),(za(21),tana4a),
     &  (za(22),tana5a),(za(23),tana6a),(za(24),la),(za(25),phia)
      double precision hb,alb,vb,slb,wlb,c1b,s2b,w2b,c2b,s3b,w3b,
     &  xb,yb,x1b,x2b,x3b,y1b,y2b,y3b,
     &  tana2b,tana3b,tana4b,tana5b,tana6b,lb,phib
      equivalence (zb(1),hb),(zb(2),alb,vb),
     &  (zb(3),slb),(zb(4),wlb),(zb(5),c1b),
     &  (zb(6),s2b),(zb(7),w2b),(zb(8),c2b),
     &  (zb(9),s3b),(zb(10),w3b),(zb(11),xb),(zb(12),yb),
```

```
     &  (zb(13),x1b),(zb(14),x2b),(zb(15),x3b),
     &  (zb(16),y1b),(zb(17),y2b),(zb(18),y3b),
     &  (zb(19),tana2b),(zb(20),tana3b),(zb(21),tana4b),
     &  (zb(22),tana5b),(zb(23),tana6b),(zb(24),lb),(zb(25),phib)
      double precision coll,slp,frct,c3,s4,w4,x4,y4,tana7,tana8,l,
     &  h,phih,rtot,xtot,ztot,do
      equivalence (z(51),coll),(z(52),slp),(z(53),frct),(z(54),c3),
     &  (z(55),s4),(z(56),w4),(z(57),x4),(z(58),y4),
     &  (z(59),tana7),(z(60),tana8),(z(61),l),
     &  (z(62),h),(z(63),phih),
     &  (z(64),rtot),(z(65),xtot),(z(66),ztot),(z(67),do)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      double precision delyk,twod,halfd,dsq
      common /VANCH/ delyk,twod,halfd,dsq
*******************************************************************************
      cosdp=dcos(phia-phip)
      if ( not ((isol eq 3  and  ibrnch eq 1)  or  isol eq 4)) goto 510
      call GCOEFF(nca,za,tnafa,ga11,ga12,ga21,ga22,ga31,ga32,2)
      xfa=one/cosdp
  510 continue
      tana=tnafa/cosdp
      lla=la*dsqrt(cosdp*cosdp+tnafa*tnafa)/SECNT(tnafa)
      xxla=halfd*dsin(phip)
      temp=xxla+xla*cosdp
      yy7=half*delyk+yla
      if (nca eq 1) goto 520
      xx3a=temp
      temp=temp+x2a*cosdp
      yy7=yy7+y2a
      if (nca eq 2) goto 520
      xx5a=temp
      temp=temp+x3a*cosdp
      yy7=yy7+y3a
  520 continue
      xx7=temp

      cosdp=dcos(phib-phip)
      if ( not ((isol eq 3  and  ibrnch eq 2)  or  isol eq 4)) goto 530
```

```
      call GCOEFF(ncb,zb,tnafb,gb11,gb12,gb21,gb22,gb31,gb32,2)
      xfb=one/cosdp
  530 continue
      tanb=tnafb/cosdp
      llb=lb*dsqrt(cosdp*cosdp+tnafb*tnafb)/SECNT(tnafb)
      if (ncb eq 1) goto 540
      xx3b= -xx1a+x1b*cosdp
      if (ncb eq 2) goto 540
      xx5b=xx3b+x2b*cosdp
  540 continue

      yy7=zero
      cosdp=dcos(ph1h-ph1p)
      if (1 gt zero) goto 550
      tna=tana7
      x=zero
      y=yy7
      goto 555
  550 continue
      tna=tnafh
      x=l*csafh
      y=yy7+l*snafh
  555 continue
      x=zero
      sca=SECNT(tna)
      g1=(tna+sca)*dexp(-w4*x/h)
      g2=y-(h*sca/w4)
      xf=one/cosdp
      tanr=tnafh/cosdp
      ll=l*dsqrt(cosdp*cosdp+tnafh*tnafh)/scafh
      xx8=xx7+x4*cosdp

      return
      end
*
```

```
et sys final/f2for/cten2 for##
      subroutine CTEN2(nc,z,vc0,th,tanb,ang,ten,vten)
*****************************************************************************
      implicit double precision (a-z)

      integer*2 nc
      double precision z(25),vc0(6),th,tanb,ang(6),ten(6),vten(6)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      integer*2 ic,in,j
*****************************************************************************
      sinb=tanb/SECNT(tanb)
      bdeg=datan(tanb)*raddeg

      do 100 ic=1,nc
      do 100 j=1,2
      in=2*(ic-1)+j
      t=(vc0(in)-th)*1.0d-3
      if (t .lt. zero) goto 20
      ten(in)= -1*sinb
      vten(in)=sinb*ten(in)
      ang(in)=bdeg
      goto 50
   20 continue
      ten(in)= -1
      vten(in)= -1
      ang(in)=90.0
   50 continue
  100 continue

      return
      end
*
```

```
el sys final/f2for/cten3 for##
      subroutine CTEN3(nc,z,tanb,tx,ang,ten,vten)
*****************************************************************************
      implicit double precision (a-z)

      integer*2 nc
      double precision z(25),tanb,tx,ang(6),ten(6),vten(6)

      double precision pi,halfpi,degrad,raddeg,zero,one,half
      integer*2 izero,ione,itwo
      common /VCONST/ pi,halfpi,degrad,raddeg,zero,one,half,
     &  izero,ione,itwo

      integer*2 ic,in,j
*****************************************************************************
      ten1(w)=hsecb-w*sinb

      h=z(1)
      s1=z(3)
      w1=z(4)
      c1=z(5)
      s2=z(6)
      w2=z(7)
      c2=z(8)
      s3=z(9)
      w3=z(10)
      l=z(24)

      secb=SECNT(tanb)
      sinb=tanb/secb
      b=datan(tanb)
      hsecb=h*secb

      ang(1)=datan(z(2))
      ang(2)=datan(z(19))
      if (nc eq 1) goto 1000
      ang(3)=datan(z(20))
      ang(4)=datan(z(21))
      if (nc eq 2) goto 1000
      ang(5)=datan(z(22))
      ang(6)=datan(z(23))
 1000 continue
```

```
      if (l eq 0 0d0) goto 1810
      if (l ge s1) goto 1300
      ten(1)=ten1(w1*l)
      goto 1820

 1300 continue
      if (l gt s1) goto 1400
      if (nc eq 1) goto 1320
      tf3=h*SECNT(z(20))*dcos(ang(3)-b)
      ten(1)=tf3-(w1*s1+c1)*sinb
      ten(2)=tf3-c1*sinb
      goto 1350
 1320 continue
      ten(1)=tx-w1*s1*sinb
      ten(2)=tx
 1350 continue
      goto 1830

 1400 continue
      if (l ge s1+s2) goto 1500
      wgt2=w2*(l-s1)
      ten(1)=ten1(w1*s1+c1+wgt2)
      ten(2)=ten1(c1+wgt2)
      ten(3)=ten1(wgt2)
      goto 1840

 1500 continue
      if (l gt s1+s2) goto 1600
      if (nc eq 2) goto 1520
      wgt2=w2*s2+c2
      tf5=h*SECNT(z(22))*dcos(ang(5)-b)
      ten(1)=tf5-(w1*s1+c1+wgt2)*sinb
      ten(2)=tf5-(c1+wgt2)*sinb
      ten(3)=tf5-wgt2*sinb
      ten(4)=tf5-c2*sinb
      goto 1550
 1520 continue
      wgt2=w2*s2
      ten(1)=tx-(w1*s1+c1+wgt2)*sinb
      ten(2)=tx-(c1+wgt2)*sinb
      ten(3)=tx-wgt2*sinb
      ten(4)=tx
 1550 continue
```

```
      goto 1850

 1600 continue
      if (l.ge.s1+s2+s3) goto 1700
      wgt3=w3*(1-s1-s2)
      wgt2=c1+w2*s2+c2+wgt3
      ten(1)=ten1(w1*s1+wgt2)
      ten(2)=ten1(wgt2)
      ten(3)=ten1(wgt2-c1)
      ten(4)=ten1(c2+wgt3)
      ten(5)=ten1(wgt3)
      goto 1800

 1700 continue
      wgt3=w3*s3
      wgt2=c1+w2*s2+c2+wgt3
      ten(1)=tx-(w1*s1+wgt2)*sinb
      ten(2)=tx-wgt2*sinb
      ten(3)=tx-(wgt2-c1)*sinb
      ten(4)=tx-(c2+wgt3)*sinb
      ten(5)=tx-wgt3*sinb
      ten(6)=tx
      goto 1900

 1810 continue
      ten(1)=h*SECNT(z(2))
 1820 continue
      ten(2)=h*SECNT(z(19))
 1830 continue
      if (nc.eq.1) goto 1900
      ten(3)=h*SECNT(z(20))
 1840 continue
      ten(4)=h*SECNT(z(21))
 1850 continue
      if (nc.eq.2) goto 1900
      ten(5)=h*SECNT(z(22))
 1860 continue
      ten(6)=h*SECNT(z(23))
 1900 continue

      do 2010 ic=1,3
      do 2010 j=1,2
      in=2*(ic-1)+j
```

```
      ten(in)=ten(in)*1 0d-3
      vten(in)=ten(in)*dsin(ang(in))
      ang(in)=ang(in)*raddeg
 2010 continue
      return
      end
*
```

END

DATE
FILMED

5 83

DTI